Containerlogistik

Aufgaben

Auf großen Frachtschiffen können bis zu 24000 Container transportiert werden. Die Verladung der Container wird mit einem objektorientierten Softwaresystem unterstützt. Zur Organisation des Frachttransports wird ein Datenbanksystem eingesetzt.

1 Im Hafen werden die Container im sogenannten Blocklager gelagert. Dort können die Container von voll automatisierten Kransystemen, sogenannten Containerbrücken, nach Bedarf umgeladen und sortiert werden. Auf einem Containerschiff werden die Container übereinandergestapelt. Material 1 zeigt das Stauprinzip eines Containerschiffs.

1.1 Das UML-Klassendiagramm in Material 2 zeigt ein Modell des Softwaresystems. Nennen Sie die im UML-Klassendiagramm auftretenden Beziehungstypen und erläutern Sie die Bedeutung jedes Beziehungstyps anhand eines Beispiels aus dem vorliegenden Diagramm sowie die Implementierung von Beziehungen in einer objektorientierten Programmiersprache.

(6 BE)

1.2 Das Containerschiff Flying Dutchman besitzt 20 Bays mit jeweils bis zu 13 Tiers. In den leeren Stack der Row 10 in Bay 02 werden für den Zielhafen Rotterdam (NLROT) nacheinander die Container mit den BIC-Codes ABMU3301605, QSHI8339397 und TFGU1849355 geladen. Entwickeln und zeichnen Sie ein Objektdiagramm für dieses Szenario.

(7 BE)

1.3 Überführen Sie die Klasse `Stack` aus dem UML-Klassendiagramm (Material 2) in Anweisungen einer objektorientierten Programmiersprache und implementieren Sie alle Methoden.

Hinweis: Die Hinweise zum UML-Klassendiagramm sind zu beachten.

(16 BE)

1.4 Jeder Container erhält einen weltweit eindeutigen BIC[1]-Code, der an beiden Stirnfronten deutlich sichtbar angebracht wird. Er besteht aus dem vierstelligen Owner-Code (Buchstaben A–Z), der den Eigentümer eindeutig identifiziert, einer sechsstelligen Seriennummer sowie einer aus Owner-Code und Seriennummer errechneten Prüfziffer.

Beispiel BIC-Code:

C S Q Z 3 0 5 4 3 8 3

Owner Code (C S Q Z) – Seriennummer (3 0 5 4 3 8) – Prüfziffer (3)

Zur Berechnung der Prüfziffer werden der Owner-Code und die Seriennummer in eine Folge von 10 Zahlen umgewandelt. Dazu wird jedem Buchstaben des Owner-Codes ein äquivalenter Zahlenwert zugeteilt. Der Buchstabe A erhält den Wert 10. Unter Auslassung der 11 und deren Vielfachen wird für die Buchstaben des Alphabets weiter gezählt (B = 12, C = 13 usw.). Die Ziffern der Seriennummer werden als separate Zahlen betrachtet. Anschließend wird jede Zahl

[1] Organisation B.I.C. (Bureau International des Containers et du Transport Intermodal).

der Folge entsprechend ihrer Position i mit 2^i multipliziert. Die Positionen sind nullbasiert. Die Ergebnisse der Multiplikationen werden summiert. Die Summe wird durch 11 geteilt und das dezimale Ergebnis zur vollen Zahl abgerundet. Die abgerundete Zahl wird dann mit 11 multipliziert und von der zuvor berechneten Summe abgezogen. Die resultierende Differenz ist die Prüfziffer. Falls die Differenz den Wert 10 hat, lautet die Prüfziffer 0.
Entwickeln und zeichnen Sie für die Methode `berechnePruefziffer()` der Klasse `Container` ein Struktogramm, das das beschriebene Berechnungsverfahren darstellt. Eine Beispielrechnung finden Sie in Material 3.

Hinweis: Die Berechnung des äquivalenten Zahlencodes für die Buchstaben des Owner-Codes ist nicht abzubilden.

(8 BE)

1.5 Für die Containerverladung wurde ein UML-Anwendungsfalldiagramm (Material 4) entwickelt. Geben Sie den Zweck von UML-Anwendungsfalldiagrammen an. Beschreiben Sie anhand des dargestellten Diagramms die wesentlichen Notationselemente sowie die Unterschiede zwischen extend- und include-Beziehungen.

(5 BE)

1.6 Die Beladung von Containerschiffen erfolgt unter der Prämisse, dass auf der Transportroute des Schiffs beim Laden und Entladen der Container in den Zielhäfen keine Container umgepackt werden sollen. Das Blocklager (Material 2) erstellt anhand der Schiffsroute eine Stauliste für das Beladen, in der die Container in umgekehrter Reihenfolge der Zielhäfen der Route einsortiert werden.
Implementieren Sie die Methode `erstelleStauliste()` der Klasse `Blocklager`.

Hinweis: Die Dokumentation der Klassen `List` und `String` finden Sie in Material 5.

(8 BE)

1.7 Die Methode `staueContainer()` der Klasse `ContainerSchiff` lädt die Container aus einer Stauliste, die das Blocklager mit der Methode `erstelleStauliste()` erzeugt, auf das Schiff. Der Ladevorgang beginnt in der angegebenen Bay. Die Stacks einer Bay werden der Reihe nach mit den Containern der Stauliste gefüllt. Falls eine Bay gefüllt ist und die Stauliste noch weitere Container enthält, wird die Beladung in der nächsten freien Bay fortgesetzt. Container, die nicht verladen werden können, werden von der Methode zurückgegeben.
Implementieren Sie die Methode `staueContainer()`.

Hinweis: Die Dokumentation der Klasse `List` finden Sie in Material 5.

(10 BE)

2 Nachfolgend soll eine Datenbank zur Unterstützung der internationalen Containerlogistik entwickelt werden. Große Seehafenstädte besitzen mehrere Containerterminals, über die die Logistik abgewickelt wird. Ein Terminal verfügt über mehrere Liegeplätze, für die jeweils mehrere fahrbare Containerbücken bereitstehen, welche die Be- und Entladung der Schiffe übernehmen.

2.1 Die Tabelle in Material 6 zeigt einen Auszug der Liegeplatzbelegungen in Hamburg. Beschreiben Sie mögliche Probleme, die sich aus der Tabellenstruktur ergeben und erläutern Sie an Beispielen aus der Tabelle die Begriffe Einfüge-, Änderungs- und Löschanomalien.

(5 BE)

2.2 Entwickeln und zeichnen Sie auf Basis der Tabelle in Material 6 ein Entity-Relationship-Diagramm mit den Entitätstypen Containerschiff, Reederei, Route, Hafen, Terminal und Liegeplatz.

Hinweis: Das Diagramm ist mit Entitätstypen, Attributen und Beziehungen mit Kardinalitäten in der [min, max]-Form darzustellen.

(9 BE)

2.3 Überführen Sie die Tabelle der Liegeplatzbelegungen (Material 6) in ein relationales Tabellenmodell in der 3. Normalform und begründen Sie ihre Entscheidungen.

Hinweis: Alle Relationen sind in der Schreibweise `Relation(PK, Attribut, ..., FK#)` anzugeben.

(8 BE)

2.4 Einige Daten der Datenbank sollen ausgewertet werden.

2.4.1 Geben Sie eine SQL-Anweisung an, die die Namen, IMO-Nummern und Containerkapazitäten aller Schiffe der Reederei Marsek ausgibt.

(3 BE)

2.4.2 Formulieren Sie eine SQL-Anweisung, die alle Schiffe ermittelt, die vom 20.03.2022 bis zum 25.03.2022 am Terminal Altenwerder abgefertigt worden sind.

(4 BE)

2.4.3 Für ein Containerschiff wird am 20.03.2022 in der Zeit von 10:00 Uhr bis 22:00 Uhr ein freier Liegeplatz im Hamburger Hafen gesucht.
Implementieren Sie eine SQL-Anweisung, die für diesen Zeitraum eine Liste aller freien Liegeplätze ausgibt.

Hinweis: Die Liste soll jeweils den Terminalnamen und die Liegeplatzbezeichnung enthalten und alphabetisch sortiert sein.

(6 BE)

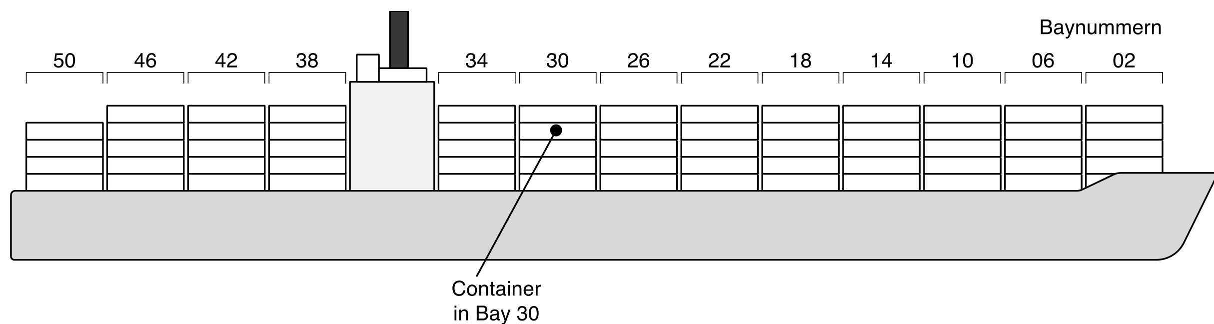
2.4.4 Die Verwaltung des Rotterdamer Hafens benötigt eine Übersicht der prozentualen Auslastung aller Liegeplätze im Jahr 2021, die jeweils den Namen des Terminals, die ID und die Bezeichnung des Liegeplatzes, die Belegung in Stunden sowie die prozentuale Auslastung in absteigender Reihenfolge der Auslastung enthält.
Entwickeln Sie eine SQL-Anweisung zur Erstellung dieser Übersicht.

Hinweise: Berücksichtigen Sie nur Liegezeiten, die vollständig im Jahr 2021 liegen. Die Methode `TIMEDIFF(a, b)` berechnet die Zeitdifferenz zwischen den `DateTime`-Werten `a` und `b` im Format `hh:mm:ss`. Die Methode `TIME_TO_SEC(x)` wandelt den Zeitwert `x` in Sekunden um.

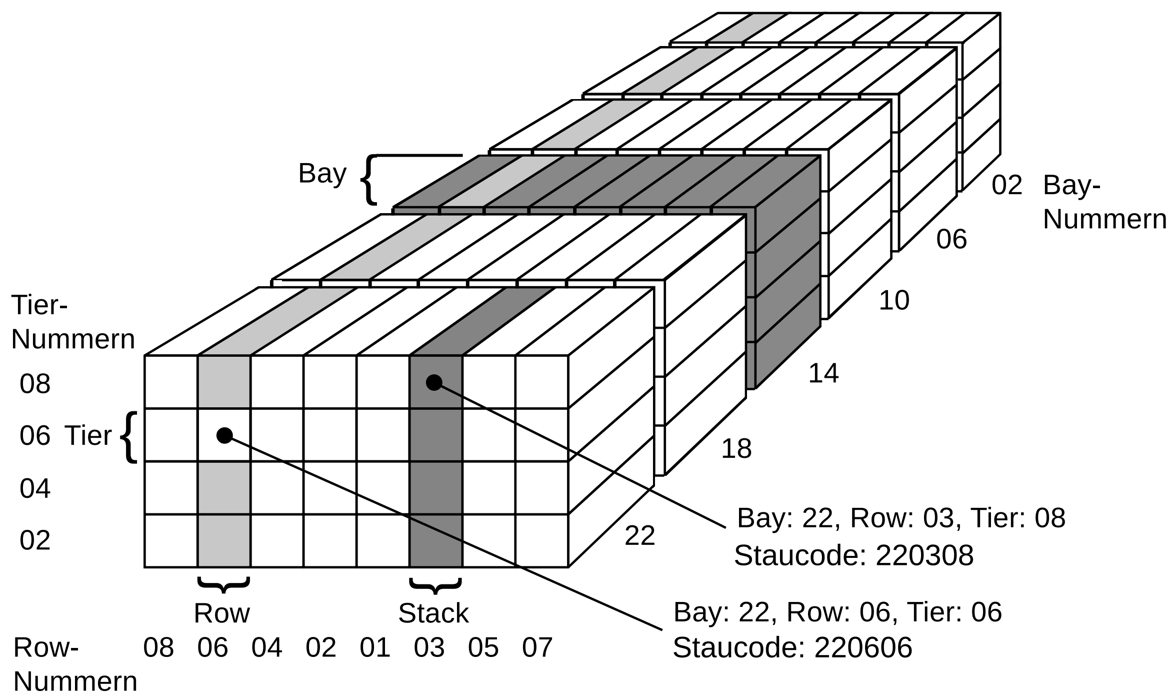
(5 BE)

Material 1

Containerschiff mit Stauplan



Die Containerstapel werden in sogenannten Bays nebeneinander angeordnet. Die Bays werden beginnend vom Bug aufsteigend nach dem abgebildeten Prinzip nummeriert.

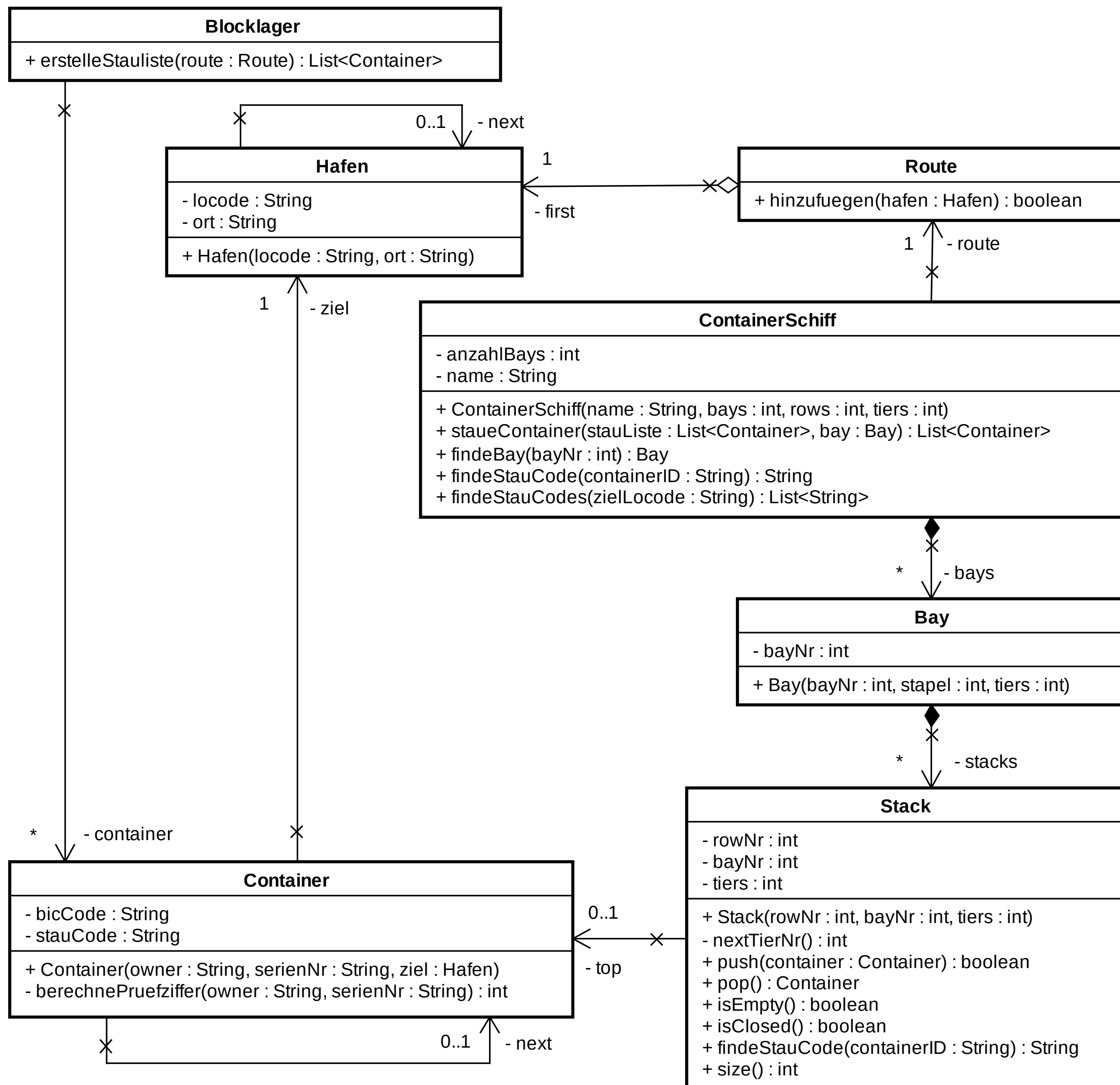


Die Container werden in Stacks übereinander gestapelt. Auf Basis eines numerischen Koordinatensystems aus Bay-, Row- und Tier-Koordinaten erhält jeder Container beim Verladen einen sechsstelligen Staucode, der sich aus den jeweils zweistelligen Angaben seiner Bay-, Row- und Tierkoordinaten zusammensetzt und seinen Stauplatz eindeutig lokalisiert.

Die Tiernummern werden, beginnend mit 02, in Zweierschritten hochgezählt. Die backbordseitigen Reihen erhalten gerade Row-Nummern, die steuerbordseitigen ungerade.

Material 2

UML-Klassendiagramm



Hinweise:
Klasse `Bay`: Der Parameter `stapel` gibt die Anzahl der Stacks einer Bay an.
Klasse `Stack`:

- `tiers` gibt die höchstmögliche Anzahl der Tiers der Stacks an. Alle Stacks haben dieselbe Größe.
- Die Methode `nextTierNr()` berechnet die Tiernummer der nächsten Ebene.
- Die Methode `push()` liefert `true`, wenn der Container auf den Stack gepackt werden konnte, sonst `false`. Sie erzeugt außerdem den sechsstelligen Staucode des Containers (z.B. 020508).
- Die Methode `pop()` entfernt den obersten Container vom Stack und gibt ihn zurück.
- Die Methode `isClosed()` gibt `true` zurück, wenn der Stack vollständig gefüllt ist.
- Die Methode `size()` gibt die Anzahl der Container in einem Stack zurück.
- Die Methode `findeStauCode()` sucht anhand der `containerID` nach dem zugehörigen Container und gibt dessen Staucode zurück oder bei Misserfolg `null`.

Auf alle Attribute kann mit get- und set-Methoden zugegriffen werden.

Material 3

Beispiel Prüfziffer einer Container-ID berechnen

1. Owner-Code und Seriennummer werden in Zahlenfolge umgewandelt.
   TFGU 411222 → 31, 16, 17, 32, 1, 1, 2, 2, 2

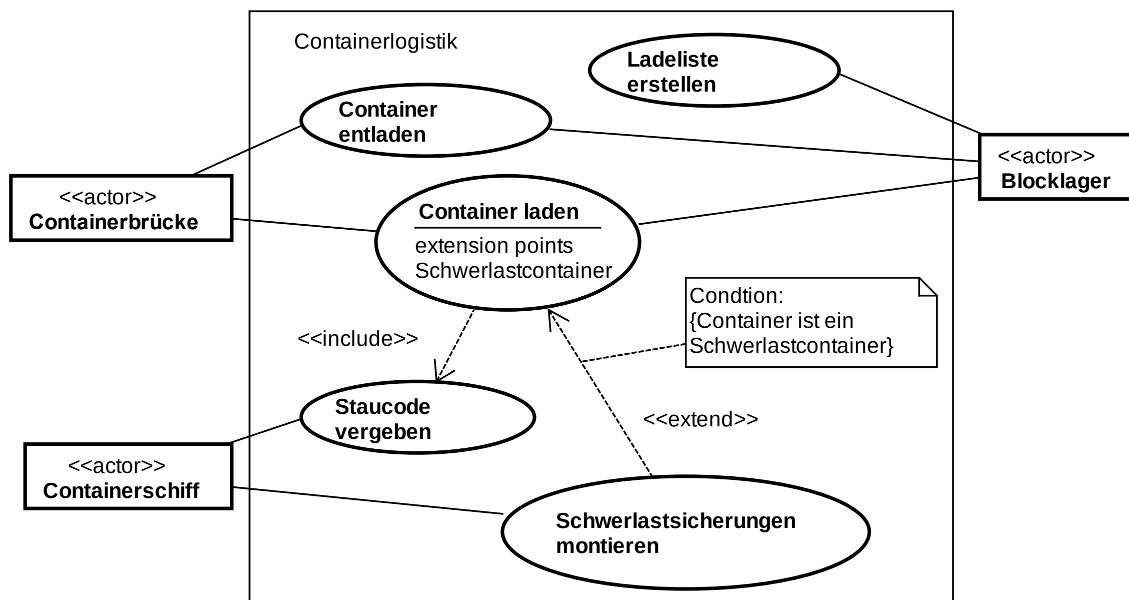
2. Zahlen werden der Reihe nach mit Potenzen von 2^0 bis 2^9 multipliziert und anschließend addiert.
   $31 \cdot 2^0 + 16 \cdot 2^1 + 17 \cdot 2^1 + 16 \cdot 2^2 + \ldots + 2 \cdot 2^9 = 2339$

3. Die Summe wird durch 11 dividiert und anschließend der ganzzahlige Anteil des Ergebnisses mit 11 multipliziert.
   2339 : 11 = 212,6…
   212 · 11 = 2332

4. Die Prüfziffer ist die Differenz aus der Summe aus 2. und dem Ergebnis aus 3.
   2339 – 2332 = 7

Material 4

Anwendungsfalldiagramm



Material 5

Klassendokumentationen

Klasse `List`

List<T>
+ List<T>() + add(obj : T) : void + add(index : int, obj : T) : void + addAll(objects : List<T>) : void + contains(obj : int) : boolean + get(index : int) : T + remove(index : int) : T + remove(obj : T) : boolean + size() : int

`List<T>()`
erzeugt eine generische Liste mit Elementen des Typs `T`.

`add(obj: T)`
hängt das Objekt `obj` vom Typ `T` am Ende der Liste an.

`add(index: int, obj: T)`
fügt das Objekt `obj` vom Typ `T` an der Position `index` in die Liste ein.

`addAll(objects: List<T>)`
hängt die Objekte `objects` vom Typ `T` am Ende der Liste an.

`contains(obj: T): boolean`
liefert `true`, wenn das Objekt `obj` in der Liste enthalten ist, ansonsten `false`.

`get(index: int): T`
liefert das Listenelement an der Position `index` zurück bzw. `null`, falls `index` negativ oder größer gleich der Anzahl der momentan enthaltenen Elemente ist.

`remove(index: int): T`
entfernt das Objekt vom Typ `T` an der Position `index` aus der Liste und gibt es zurück.

`remove(obj: T): boolean`
entfernt das Objekt `obj` aus der Liste. Falls `obj` mehrmals in der Liste enthalten ist, wird nur das erste Vorkommen entfernt. Der Rückgabewert ist `true`, falls das Objekt gefunden und entfernt wurde, sonst `false`.

`size(): int`
liefert die Anzahl der Elemente in der Liste zurück.

Klasse `String`

String
+ equals(str : String) : boolean + split(str : String) : String[ ] + startsWith(str : String) : boolean

`equals(str: String): boolean`
liefert `true`, wenn beide Strings gleich sind, andernfalls `false`.

`split(str: String): String[]`
teilt einen String am Trennzeichen `str`. Die Teil-Strings werden in einem Feld zurückgeliefert.

`startsWith(str: String): boolean`
liefert `true`, wenn der String mit `str` beginnt, andernfalls `false`.

Material 6

Tabelle der Liegeplatzbelegungen in Hamburg (Auszug)

Schiff	IMO-Nr.	Con-tainer-kapa-zität	Reederei	Route	Terminal	Liege-platz	Beginn	Ende
Globetrotter	945920	24000	Marsek, Kopen-hagen	1. Rotterdam (NLROT), 2. Shanghai (CNSHA), 3. Singapur (SGSIN)	Altenwerder, Am Ballinkai 1, 21129 Hamburg	A3	26-01-2022 18:30:00	27-01-2022 20:00:00
…	…		…	…	…	…	…	…
Oceanrider	549234	21000	Marsekk, Kopen-hagen	1. Rotterdam (NLROT), 2. Shanghai (CNSHA), 3. Singapur (SGSIN)	Altenwerder, 21129 Hamburg	A2	20-03-2022 10:30:00	21-03-2022 18:00:00
Seamaster	665718	19000	Toledo, Lissabon	1. Shanghai (CNSHA), 2. Singapur (SGSIN)	Burchardkai, Waltershoferdamm, 21129 Hamburg	B2	20-03-2022 12:00:00	20-03-2022 22:00:00
Globetrottel	945920	24000	Marsek, Kopen-hagen	1. Antwerpen (BEANR), 2. Lissabon (PTLIS)	Tollerort, Am Vulkanhafen 30, 20457 Hamburg	T1	20-03-2022 16:00:00	22-03-2022 08:00:00
…	…		…	…	…	…	…	…

Hinweise: IMO-Nr. ist eine international eindeutige Kennnummer für Containerschiffe, die von der International Maritime Organisation vergeben wird. Jeder Hafen wird mit einem Location-Code, der sich aus Kürzeln für das Land und die Stadt zusammensetzt, eindeutig identifiziert. Beginn und Ende der Liegezeit sind im `DateTime`-Format (`dd-mm-yyyy hh:mm:ss`) angegeben.